IBL Digital Signal Processor Design Project

Team DIC-sie Chicks

Sam Schulman
ECE 3663 – Spring 2013
University of Virginia
sps4zp@virginia.edu

Evelyn Ponce
ECE 3663 – Spring 2013
University of Virginia
eip3uc@virginia.edu

Chaitanya Bhave
ECE 3663 – Spring 2013
University of Virginia
cmb5eg@virginia.edu

1. Introduction

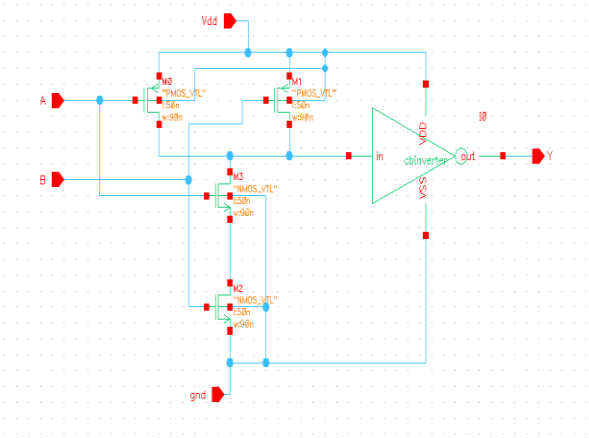
In this paper, Team DIC-sie Chicks will discuss our development of an embedded Digital Signal Processor, designed in the FreePDK 45nm technology. This processor aims to combine efficiency with utility in an all-new design that takes processor innovation back to its roots, where simplicity is the best solution. Designed for ease of use by corporations and individuals with a wide variety of educational backgrounds, this Digital Signal Processor is easy to understand and easy to fix. Its easy-to-follow design and thorough accompanying schematics allow for quick comprehension and visualization of the operation of the total unit.

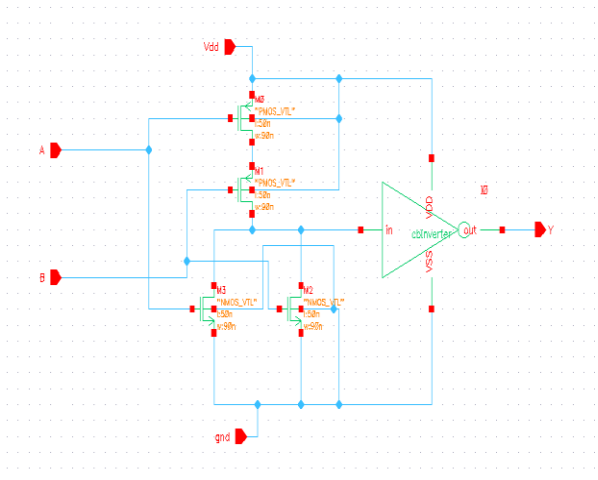
2. Design Description

This Digital Signal Processor effectively combines both utility and efficiency. With low power consumption, this unit offers a variety of standard functions for 8-bit inputs, including And, Or, 2's Complement, Add, Pass A, and NOP, with an additional, optional Logical Shift Left function also available. This Digital Signal Processor also gives the option of adding a further component of your choice, as it has an available port for one additional connection. This design aims to optimize the user-friendliness while retaining low values of worst-case delay, total area, and energy consumption. It also has a low IBL design metric, defined by the equation: (Active Power)*Delay^2*Area.

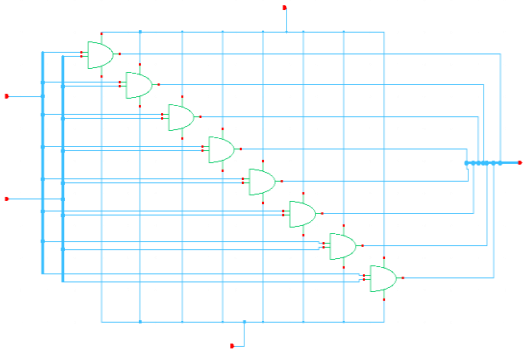
In order to minimize testing issues, all base-design functions were generated in 1-bit versions first. These smaller versions were then tested and iterated to create the 8-bit components required for the ALU. As a result, it was simpler to identify and correct errors in the base schematics. This also allowed for greater ease in observing the behavior of the 1-bit pieces of each function, which provided another method of ensuring the functions were actually serving their intended purpose.

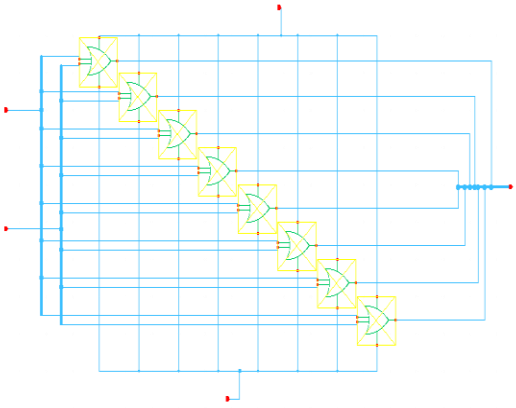
To construct the AND and OR gates, a 1 bit, 2 input gate was made of each.





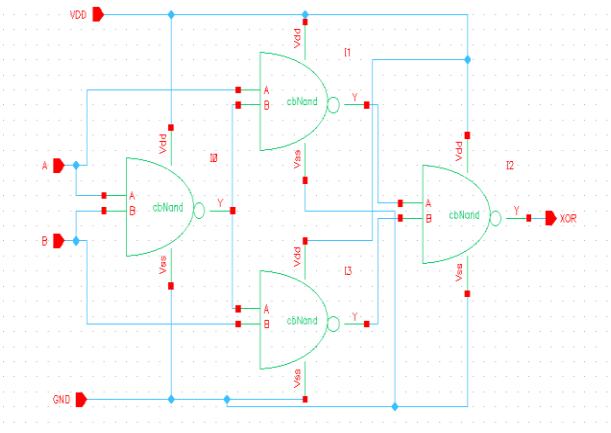
Then, the created symbol was duplicated 8 times and connected to construct 8 bit gates as follows.

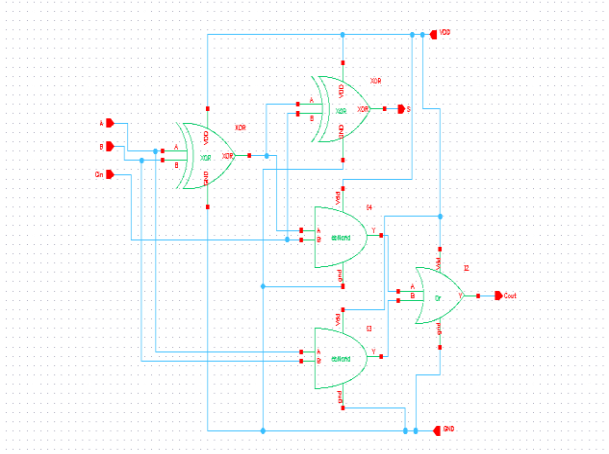


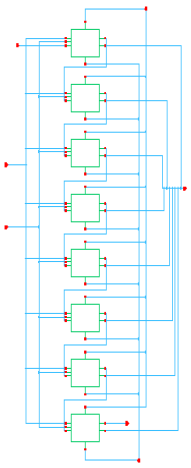


A symbol of the final 8 bit gates was also created.

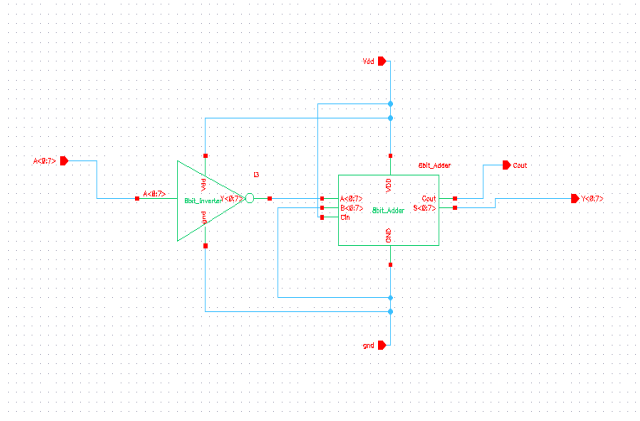
Team DIC-sie Chicks also created an 8-bit adder, an 8-bit 2's complement, and an 8-bit register. The 8-bit adder was created from 8 one-bit full adders, with each carry-out bit connected to the next adder's carry-in bit. The one-bit full adder schematic consists of AND, OR, and XOR gates, in which the XOR gate was constructed from previously built NAND gates.



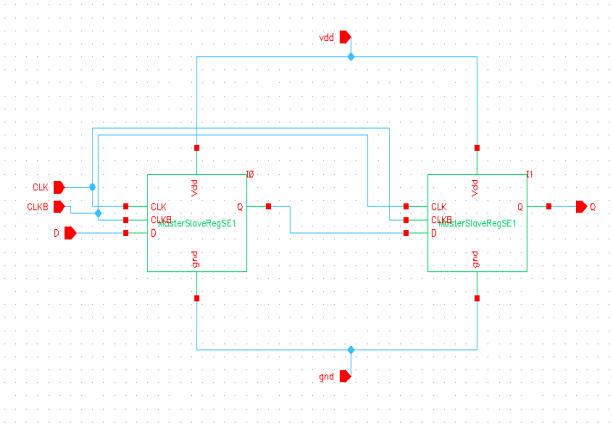




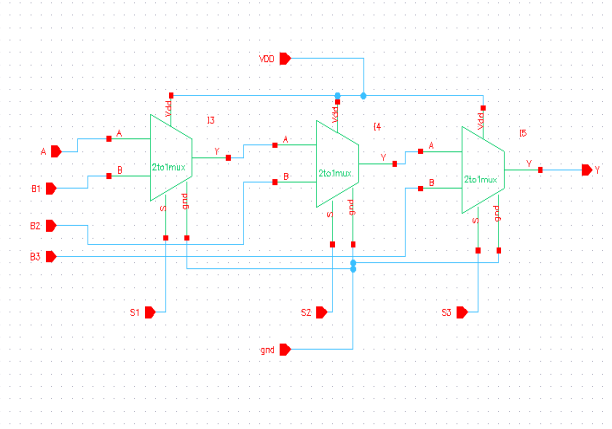
The 2's complement was put together using the 8-bit inverter and the 8-bit adder, where the A input was inverted and passed to the A input of the adder, while the B input of the adder was grounded, and the carry-in was set to high.



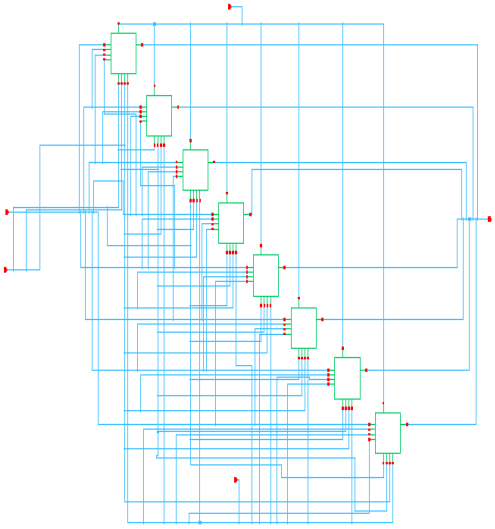
The input and output registers were created using an inverted version of the circuit for the negated-edge-sensitive D-flip-flop master-slave register schematic, as given below.



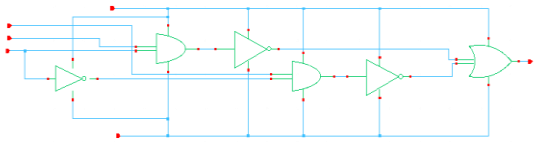
These components were combined to create the base design of the ALU. The optional shifter was put together using 8 smaller shifter units, each designed using three 2-to-1 MUXes.



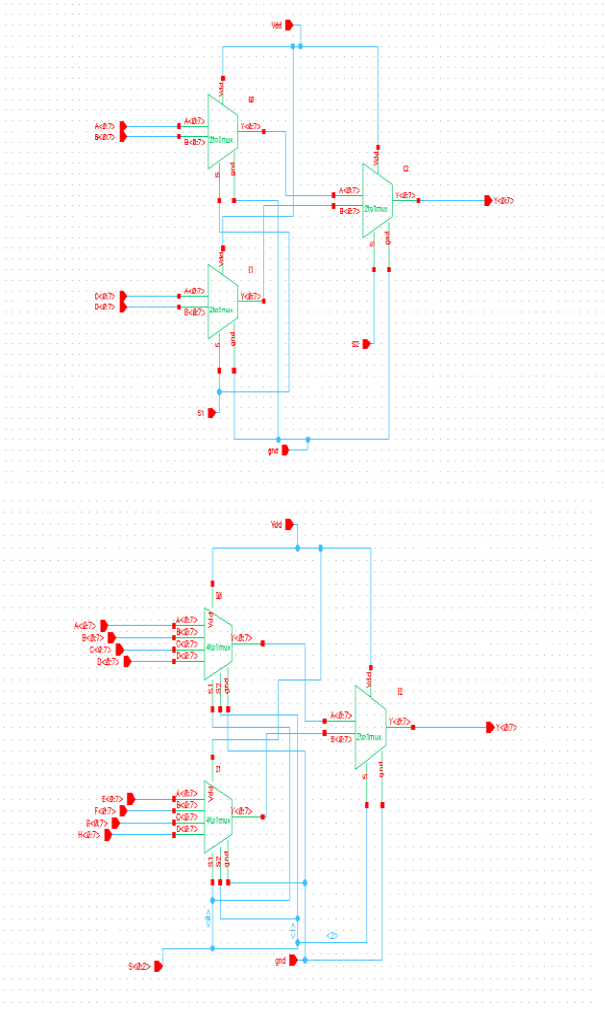
Eight of these 1-bit Shifters were then connected together in order to generate an 8-bit Shifter.



An 8:1 MUX was created using seven 2:1 MUXes, in order to provide a function-selecting mechanism.



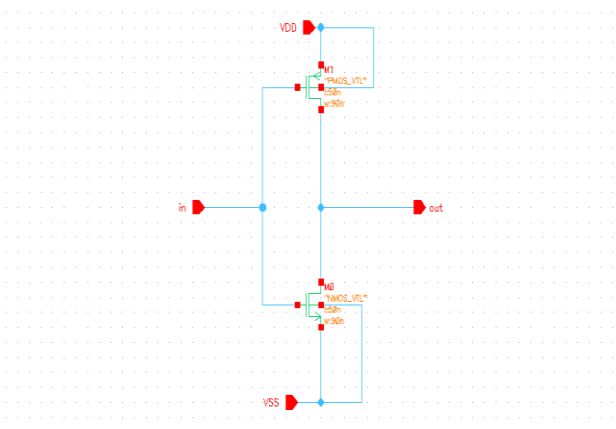
We made the decision to create an 8:1 MUX by using two 4:1 MUXes and a 2:1 MUX to combine the two outputs. Therefore we constructed a 2:1 MUX and combined two 2:1 MUXes to create the 4:1 MUX. After that, we routed the two outputs from the two 2:1 MUXes used to create the 4:1 MUX through yet another 2:1 MUX. This generated the final 8:1 MUX design, which employed seven 2:1 MUXes.

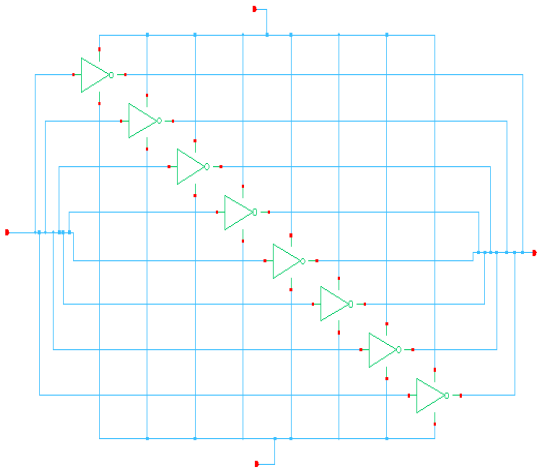


The MUX selects functions based on the 3-bit Select input, the value-to-function associations of which are given by the table below.

Select Input	ALU Functions	Description
000	AND	Out = A & B
001	OR	Out = A | B
010	PASS A	Out = A
011	ADD	Out = A + B
100	2'S COMPLEMENT	Out = Two's complement of A
101	ADDITIONAL PORT	Out = TBD
110	LOGICAL LEFT SHIFT	Out = A<<B
111	NOP	No change at Out

The logic gates were then simulated to demonstrate their success. The input in all the simulations was driven by a double inverter at the output. As such, an 8-bit inverter was also created. For this, a 1-bit inverter was designed, and was then turned into an 8-bit inverter in a manner similar to that of the 8-bit AND and OR gates.





All logic gates were now ready to be simulated. All simulations were done using ADE L environment transient simulations. For the 1-bit AND and OR gates, all four possible input combinations of 00, 01, 10, and 11 were tested by checking the transient graphs.

Delays

We used the simulations labeled below to get the worst-case delay for each operation.

Operation	Worst-Case Delay	Scenario
ADD	465.8 ps	B=00000001, "high" and A= goes "low" to" high"; 00000000 to 11111111 and carry in value has to be low
2COMP	127 ps	When all the bits are going from low to high; when A transitions from 00000000 to 11111111
AND	314.429 ps	B= 11111111; "high" and A= transitions from "low" to "high"; 00000000 to 11111111
OR	327.1 ps	A & B=00000000 to 11111111; goes from "low" to "high"
Pass A	234.22 ps	A goes from "low" to "high"; 00000000 to 11111111

Energy

In order to get the energy per operation, the approach directed to take was to use the calculator provided by Cadence after running simulations. However, we could not get the calculator to function properly in the given amount of time. We therefore calculated the energy using DC simulations of each component. A resistor was connected from the output of the given operation to ground, and then a DC simulation was run in which the current through the resistor was measured and recorded. The voltage difference of the source voltage and the resistor voltage became the voltage of the component. The operation was in series with the resistor and would therefore have the same value of current that was previously recorded. Using the equation P=VI, we could then calculate the power for each operation of the ALU.

The total average energy of the ALU was then measured using the same approach. Diagrams of the current measured for each component are attached for review.

Operation	**Energy**
ADD	3.0231E-4
2COMP	7.7198E-5
AND	4.696E-5
OR	3.0169E-4
Pass A	3.605E-6
Total ALU	6.6799E-5

3. INNOVATION

3.1 Sizing

In order to optimize the performance and metric of the ALU design, the team took certain measures. The metric is calculated with the equation (Delay^2)(Energy)(Area), so in order to optimize the metric, the area and delay needed to be minimized. To minimize the area, the widths of the transistors were set to the smallest value of 90nm (Wp=Wn=90nm).

3.2 Design Decisions

The main design decision made was to maximize the simplicity of the processor. The simplicity of the design makes it functional for many distinct situations and users, as well as easy to implement changes. Our design implements a simple full adder, as this will be the smallest in size and easiest to understand by any user. We chose simplicity as our main design goal because this would allow for more accurate and faster simulations of each additional aspect, such as calculating the delay, energy, and area.

3.3 Arbitrary Function

The arbitrary function we decided to create is an 8-bit shifter. Our original topology did not require a shifter, so we therefore decided that it was the best choice for the arbitrary function. The shifter we used is a right shifter, and used MUXes to create it. Three 2-to-1 MUXes were connected together to create a 1-bit shifter, and then 8 of these were connected for the final 8-bit shifter. We used the same 2-to-1 MUXes that were used to create the 8-to-1 MUX in the final ALU design. The arbitrary function therefore also has transistors with minimum widths of 90nm.

4. RESULTS

4.1 Metric

Metric = (Delay^2)(Energy)(Area) =
(465.8E-12)^2*(6.6799E-5)*(1.872E-4) =
2.7132E-27 m^2*s*W

4.2 Shifter Results

Energy	**Area**
1.7747E-4	5.184E-5

4.3 Delay and Energy Breakdown

In order to show how much delay and energy each component of the ALU contributes to the overall design, a table is given of the ratio of the worst case delay and energy of each function compared to the total delay and energy, respectively.

Operation	**Delay Ratio**	**Energy Ratio**
ADD	1	4.5257
2COMP	.2726	1.1557
AND	.6750	.7030
OR	.7022	4.5164
Pass A	.5028	.0539

5. CONCLUSION

In this paper, Team DIC-sie Chicks discussed our development of an embedded Digital Signal Processor, designed in the FreePDK 45nm technology. We designed an ALU, created using an 8-bit Adder, 2's Complement, AND, OR, Pass A, and Shifter. A schematic of each ALU component individually and implemented together, in addition to Cadence simulations that prove that each operation functions properly, are attached.

We have showed how this design achieves efficiency without compromising our goal of simplicity. Our processor has been designed in order for corporations and individuals, stemming from a variety of backgrounds, will find it easy to use and understand. The simplicity of this Digital Signal Processor also allows for changes and adaptations to be implemented for many uses without complications. Our design meets the requirements given by IBL, and should be chosen to be used by the company.

6. REFERENCES

Benton H. Calhoun, Design Principles for Digital CMOS Integrated Circuits, NTS Press, ISBN: 978-1-934891-14-8

Jan Rabaey, Anantha Chandrakasan, Bora Nikolic, Digital Integrated Circuits: A Design Perspective, (Second Edition) Prentice Hall. ISBN: 0-13-090996-3

http://venividiwiki.ee.virginia.edu/mediawiki/index.php/Main_Page